

UNIT IV

- 5.1 Input and output – Introduction to System. IO .namespace
- 5.2 File and folder operations
- 5.3 Stream class
- 5.4 Introduction to ADO .NET
- 5.5 Building data table
- 5.6 Data view
- 5.7 Data set
- 5.8 Data relations
- 5.9 ADO.NET managed providers – OleDb managed provider
- 5.10 SQLProvider.

5.1 Introduction to System.IO namespace

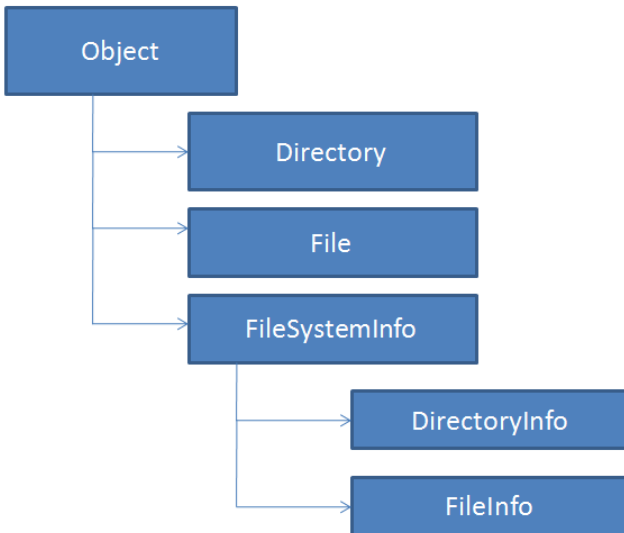
Used to manipulate the physical directories and files.
To read data from and write data to string buffers.

Members of System.IO namespace

IO Type	meaning
Directory DirectoryInfo File FileInfo	Used to manipulate the properties for a given directory or file. The Directory and File types expose their functionality as static methods. The DirectoryInfo and FileInfo types expose similar functionality as Object variable.
Path	Contains file directory path information.
StreamWriter StreamReader	Used to store text information to a file. Do not support for random access.
StringWriter StringReader	Same as StreamReader/Writer. but storage is a string buffer rather than a file.
FileStream	For random file access.
BinaryReader BinaryWriter	Used to store and retrieve primitive data types(Integer, Boolean, strings) as a binary value.

5.2 File and Folder Operations

File and Directory types



FileSystemInfo

FileSystemInfo Property	Meaning
Attributes	Gets or sets the attributes associated to current file.
CreationTime	Gets or sets the creation for the file or directory.
Exists	Used to determine if a given file or directory exists.
Extension	Used to retrieve a file extension.
FullName	Gets the full path of the directory or file.

DirectoryInfo

Members	Meaning
Create() CreateSubdirectory()	Create a directory or subdirectory.
Delete()	Deletes a directory all its contents.
GetDirectories()	Returns an array of strings to list the subdirectories.
GetFiles()	Gets the files in the specified directory.
MoveTo()	Moves a directory and its contents to a new path.

Program to display details of directory

Class direct

```
{  
    public static void Main()  
    {  
        DirectoryInfo dir=new DirectoryInfo("c:\sharp");  
        Console.WriteLine("Fullname" + dir.FullName);  
        Console.WriteLine("Creation" + dir.CreationTime);  
        Console.WriteLine("Attributes" + dir.Attributes.ToString());  
    }  
}
```

FileInfo

Used to retrieve the details of existing files.

Used to create, copy, move and delete the files.

Members of FileInfo

Members	Meaning
---------	---------

Create()	Creates the new file.
CopyTo()	Copies an existing file to a new file.
MoveTo()	Moves the file to a new location.
Delete()	Remove the file from directory.
Open()	Open the already existing file.
OpenRead()	Creates a read-only filestream.
OpenWrite()	Creates a read/write filestream.

Program to display details of the given file

```
using System;
using System.IO;
class dir
{
    public static void Main()
    {
        FileInfo f=new FileInfo("c://c#/sample1.txt");
        FileStream fs=f.Create();
        Console.WriteLine( f.CreationTime + "\n"+ f.FullName + "\n" +
            f.Attributes.ToString());
        fs.Close();
        f.Delete();
    }
}
```

FileInfo.Open()-Opens File with various read and write previlages

Syntax: FileStream f1= FileInfo.Open(FileMode.Create,FileAccess.Read,FileShare.Read);

FileMode Attribute Values

FileMode	Meaning
Append	Open the already existing file and write at the end of the file. if it not exist create the new file.

Create	Create the new file. If the file already exists, it is overwritten.
CreateNew	Create a new file. If the file already exists, an IOException is thrown.
Open	Open an existing file.
OpenOrCreate	Open an existing file; otherwise creates new file.
FileAccess	Meaning

FileAccess Attribute Values

FileAccess	Meaning
Read	Read-only access to the file.
ReadWrite	Read and write access to the file.
Write	Write access to the file.

FileShare Attribute Values

FileShare	Meaning
None	Declines sharing of the current file.
Read	Allows subsequent opening of the for reading.
ReadWrite	Allows subsequent opening of the file for reading or writing.
Write	Allows subsequent opening of the file for writing.

5.3 Stream

Members of class Stream

Stream Member	Meaning
Close()	Closes the current stream.
Length	Returns the length of the stream, in bytes.
Position	Determines the position in the current stream.
Read() ReadByte()	Reads a sequence of bytes and advances the current position.
Seek()	Sets the position in the current stream.
Write() WriteByte()	Write a sequence of bytes to the current stream and advances the current position.

5.3.1 ByteStreams

a)FileStreams – Example

```
using System;
using System.IO;
class stream
{
    public static void Main()
    {
        FileStream fs=new FileStream("test.txt",FileMode.OpenOrCreate,
                                   FileAccess.ReadWrite);

        for(int i=0; i<256; i++)
            fs.WriteByte((byte)i);
        fs.Position=0;
        for(int i=0; i<256; i++)
            Console.WriteLine(fs.ReadByte());
        fs.Close();
    }
}
```

b)Memory Streams – Example

```
using System;
```

```

using System.IO;
class stream
{
    public static void Main()
    {
        MemoryStream ms=new Memorystream();
        ms.Capacity=256;
        for(int i=0; i<256; i++)
            ms.WriteByte((byte)i);
        ms.Position=0;
        for(int i=0; i<256; i++)
            Console.WriteLine(ms.ReadByte());
        ms.Close();
    }
}

```

c)Buffered Stream – Example

```

using System;
using System.IO;
class stream
{
    public static void Main()
    {
        FileStream fs=new FileStream("test.txt",FileMode.OpenOrCreate,FileAccess.ReadWrite);
        BufferedStream bs=new BufferedStream();
        Byte[] b={0x55,0x66,0x43};
        Bs.Write(b,0,b.Length)
        bs.Close();
    }
}

```

5.3.2 Character Streams

Members of TextWriter

TextWriter Member	Meaning
Close()	Closes the writer, the buffer is automatically flushed.
Write()	Writes a line to the text stream, without a new line constant.

WriteLine()	Writes a line to the text stream, with a new line constant.
NewLine	Used to make a newline.
Flush()	Clears all buffers for the current writer.

Members of TextReader

TextReader Member	Meaning
Read()	Reads data from an input stream.
ReadBlock()	Reads a max. of count characters from the current stream and writes the data to a buffer.
ReadLine()	Reads a line of characters from the current stream and returns data as string.
ReadToEnd()	Reads all character from the current position till end and returns them as one string.
Peek()	Returns the next character without changing the position of the reader.

Stream Reader and Writer

Used to read from or write Character based data to file(e.g. string).

Object

- TextReader
 - StreamReader
 - StringReader
- TextWriter
 - StreamWriter
 - StringWriter

Example-StreamWriter

Program for Write the text content in to the file sample.txt

Class wstream

```
{
    public static void Main()
    {
```



```

        FileInfo f=new FileInfo("sample.txt");
        StreamWriter sw=f.CreateText();
        sw.WriteLine("Good Morning");
        sw.WriteLine('Have a nice day");
        sw.Close();
    }
}

```

StreamReader

Program for reading the data from sample.txt

Class sread

```

{
    public static void Main()
    {
        // create a StreamReader
        StreamReader sr=File.OpenText("sample.txt");
        string input;
        while((input=sr.ReadLine())!=null)
            Console.WriteLine(input);
        sr.Close();
    }
}

```

StringWriter and StringReader

Used to read from or writeCharacter based data to Buffer(e.g. string).

StringWriter - Example

Class strwrite

```

{
    public static void Main()
    {
        StringWriter sw=new StringWriter();
        sw.WriteLine("good morning");
        sw.WriteLine("Have a nice day");
        sw.Close();
        Console.WriteLine("contents" + sw.ToString());
    }
}

```

StringReader

Code to read the content from StringWriter:

```

    StringReader sr=new StringReader(sw.ToString());
    String input;
    While((input=sr.ReadLine())!=null)
        Console.WriteLine(input);
    Sr.Close();

```

5.3.3.BinaryReader and BinaryWriter

Used to read and write discrete data types to an underlying stream.

Members of Binary Writer

Base Stream
Close()
Flush()
Seek()
Write()

Members of BinaryReader

BaseStream
Close()
PeekChar()
Read()
ReadXXX()

Example Program

```
Using System;  
Using System.IO.*;  
Class sampleBINARY  
{  
    FileStream fs=new FileStream("temp.dat",FileMode.OpenOrCreate,FileAccess.Read);  
    BinaryWriter w=new BinaryWriter(fs);  
    w.Write(10);  
    w.Write(5.78)  
    w.Write('a');  
    w.BaseStream.Position=0;  
    BinaryReader r=new BinaryReader(fs);  
    While(r.PeekChar!=-1)  
    {  
        Console.WrieLine(r.ReadByte());  
    }  
}
```

5.4 Introduction to ADO.NET

ADO .NET is a collection of classes, interfaces, structures, and enumerated types that manage data access from relational data stores within the .NET Framework

Evolution of ADO.NET

The first data access model, DAO (data access model) was created for local databases.

Next came RDO (Remote Data Object) and ADO (Active Data Object) which were designed for Client Server architectures.

With ADO, all the data is contained in a recordset object which had problems when implemented on the network and penetrating firewalls.

ADO was a connected data access, which means that when a connection to the database is established the connection remains open until the application is closed.

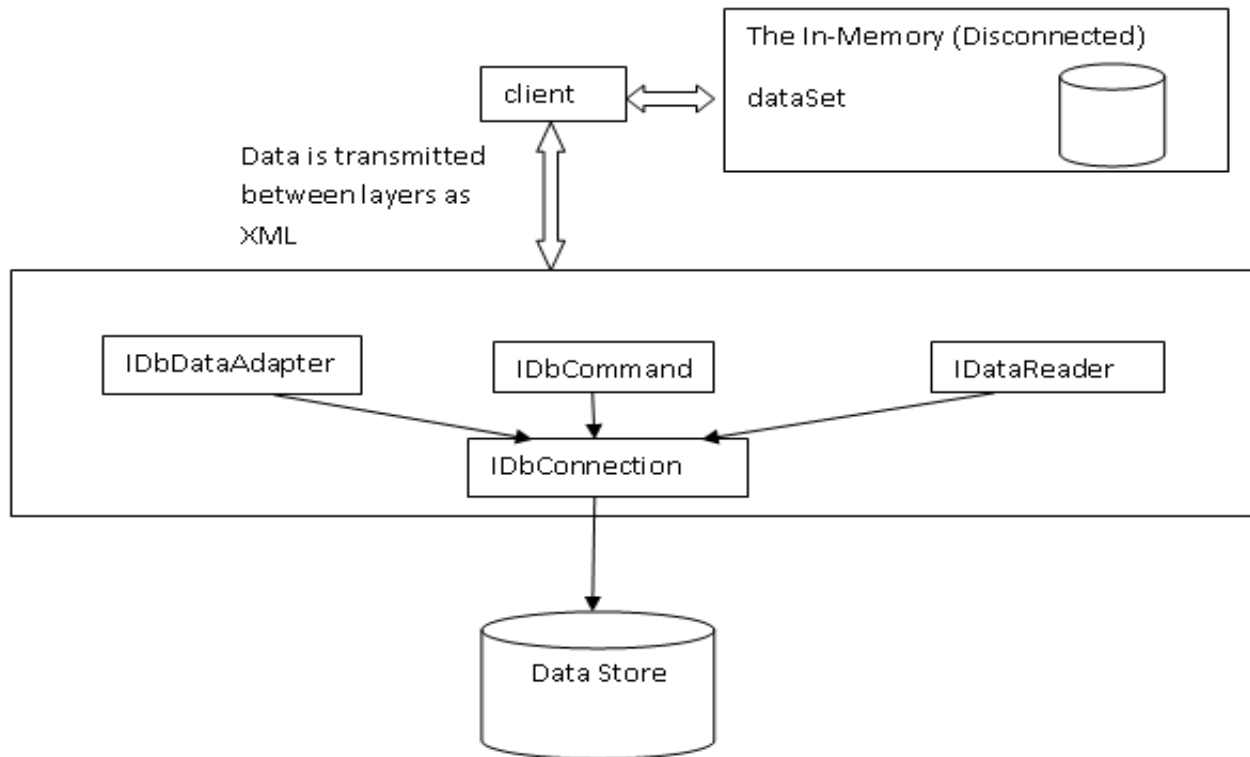
ADO .NET addresses the above mentioned problems by maintaining a disconnected database access model which means, when an application interacts with the database, the connection is opened to serve the request of the application and is closed as soon as the request is completed. If a database is updated, the connection is opened long enough to complete the update operation and is closed.

By keeping connections open for only a minimum period of time, ADO .NET conserves system resources and provides maximum security for databases and also has less impact on system performance.

ADO versus ADO .NET

Feature	ADO	ADO.NET
Primary Aim	Client/server coupled	Disconnected collection of data from data server
Form of data in memory	Uses RECORDSET object (contains one table)	Uses DATASET object (contains one or more DATATABLE objects)
Disconnected access	Uses CONNECTION object and RECORDSET object with OLEDB	Uses DATASETCOMMAND object with OLEDB
Disconnected access across multi-tiers	Uses COM to marshal RECORDSET	Transfers DATASET object via XML. No data conversions required

ADO.NET Architecture



Components

Data Access in ADO.NET relies on two components:

- DataSet
- DataProvider

DataSet

- The dataset is a disconnected, in-memory representation of data.
- It is Local copy. The data can be Manipulated and updated independent of database.

Disconnected, cached, scrollable data

DataProvider

- The Data Provider is responsible for providing and maintaining the connection to the database.
- There are two DataProviders:

SQL Data Provider (specifically for SQL Server7.0)

OleDb DataProvider (other databases like access,oracle)

Each DataProvider consists of the following component classes:

Connection object which provides a connection to the database.
 Command object which is used to execute a command.
 DataReader object which provides a forward-only, read only, connected recordset.
 DataAdapter object which populates a disconnected DataSet with data and performs update

ADO.NET Namespaces

System.Data	This namespace defines types that represent tables, rows, columns, constraints & DataSets.
System.Data.Common	This namespace contains types shared between data providers.
System.Data.OleDb	This namespace defines types that allow you to connect to an OLE.DB
System.Data.Odbc	This namespace defines types that constitute the ODBC data provider.
System.Data.OracleClient	This namespace defines types that constitute the Oracle data provider.
System.Data.SqlClient	This namespace defines types that constitute the SQL data provider.

5.5. Building Data Table

Types of System. Data

Type	Meaning
DataColumnCollection DataColumn	DataColumnCollection is used to represent all of the columns used by a given DataTable. DataColumn represents a specific column in a DataTable.
DataRowCollection DataRow	These types represent a collection of rows for a DataTable and a specific row of data in a DataTable.
DataSet	Represents an in-memory cache of data that may consist of multiple related DataTables.
DataRelationCollection DataRelation	This collection represents all relationships between the tables in a DataSet
DataTableCollection DataTable	This collection represents all of the tables for a particular DataSet.

DataColumn

1. The DataColumn type represents a single column maintained by a DataTable

2. Eg: Assume you have a table named Employees with three columns (EmpID, FirstName and LastName)

Properties of DataColumn:

Column Name	Meaning
DataType	Defines the datatype (Boolean, string, float & so on) stored in the column.
AllowDBNull	Boolean value that indicates whether the column may contain null values.
Expression	An expression defining how the value of a column is calculated.
Caption	The caption to be displayed for this column.
AutoIncrement AutoIncrementSeed AutoIncrementStep	These properties are used to configure the auto increment behaviour for a given column.
ReadOnly	Determines if this column can be modified once a row has been added to the table.
Table	Gets the DataTable that contains this DataColumn.

Enabling Auto.Incrementing Fields:-

AutoIncrementing columns are used to ensure that when a new row is added to a given table the value of this column is assigned automatically based on the current step of the incrementation.

Seed value is used to mark the starting value of the column, where step value identifies the number to add to the seed when incremented

eg:-

```
DataTable t1=new DataTable("Student");
DataColumn C1=new DataColumn();
C1.ColumnName ="ID"
C1.ColumnName=System.Type.GetType("System.Int32");
C1.AutoIncrement=true;
C1.AutoIncrementseed=1;
C1AutoIncrementStep=1;
DataColumn C2=new DataColumn();
C2.ColumnName="Name";
C2.DataType=Type.GetType("System.String");
C2.ReadOnly=True;
t1.columns.Add(C1);
```

```

t1.columns.Add(C2);

DataColumn C3=new DataColumn();
C3.ColumnName="Dept";
C3.DataType=Type.GetType("System.String");
t1.columns.Add(C1);
t1.columns.Add(C2);
t1.columns.Add(C3);

```

Data Row

- *Data Row types represents the actual data in the table.
- *cannot create direct instance of this type, but obtain reference from a given Data Table.
- *Use the method New Row() of Data Table

Members of DataRow Type:

ItemArray : This property gets or sets all of the values for this row using an array of objects.

Table : Obtaining the reference of the table containing this row.

AcceptChanges: Commit all the changes made to this row.

RejectChanges: Reject all the changes made to this row.

Delete() : Marks this row to be removed.

isNull() : Get the value indicating whether the specified column contains null

How to insert new row in the above table:

```

DataTable t1=new DataTable("Student");
//.....
.....
Creat two columns "Names" & ID.
DataRow r1=t1.NewRow();//Creat Row
r1["name"]="aaa";//insert datas
r1["Dept"]="IT";
t1.Rows.Add(r1);//Add row to the table.
DataRow r2=t1.NewRow();//Creat Row
r2["name"]="bbb";//insert datas
r2["Dept"]="IT";
t1.Rows.Add(r2);//Add row to the table.
DataRow r3=t1.NewRow();//Creat Row
r3["name"]="aaa";//insert datas
r3["Dept"]="ECE";
t1.Rows.Add(r3);//Add row to the table.

```

Data Table

Data table is an in-memory representation of a tabular block of data.

Members of data table:

Columns : Returns the collection of columns that belong to this table.
Constraints : Gets the collection of constraints maintained by the table.
DefaultView : Gets a customized view of the table.
MaximumCapacity: Gets or Sets the initial no. of rows in this table.
PrimaryKey : Gets or Sets an array of columns that function as primary keys for the table.
Rows : It returns the collection of rows that belong to this table.
TableName : Gets or Sets the name of the table.

Now the structure of Table student:

ID	Name	Dept
1	aaa	IT
2	bbb	IT
3	aaa	ECE

Building Complete DataTable:

DataTable

Create the following Data Table

ID	Name
1	aaa
2	bbb
3	aaa
4	bbb

Program to create above table

Using System.Data;

Using System;

Class sample

{

Public static void Main()

{

DataTable t1=new DataTable("Student");


```
//Create Data Column "ID" and add it to the data table.
```

```
DataColumn c1=new DataColumn();  
c1.ColumnName="ID";  
c1.DataType=Type.GetType("System.Int32");  
//Set the Auto-Increment behaviour  
c1.AutoIncrement=true;  
c1.AutoIncrementSeed=1;  
c1.AutoIncrementStep=1;  
//Add this column  
t1.Columns.Add(c1);
```

```
//Create DataColumn "Name" and add it to the table.
```

```
DataColumn c2=new DataColumn();  
c2.ColumnName= "Name";  
c2.DataType=Type.GetType("System.String");  
//add this column  
t1.Columns.Add(c2);
```

```
//Create New Row and insert datas.
```

```
DataRow r1=t1.NewRow();  
r1["Name"]="aaa";  
//Value of ID field set automatically  
//Add row to DataTable  
t1.Rows.Add(r1);  
DataRow r2=t1.NewRow();  
r2["Name"]="bbb";  
//Value of ID field set automatically  
//Add row to DataTable  
t1.Rows.Add(r2);  
DataRow r3=t1.NewRow();  
r3["Name"]="aaa";  
//Value of ID field set automatically  
//Add row to DataTable  
t1.Rows.Add(r3);  
//Create some more rows  
t1.AcceptChanges();  
}  
}
```

5.5.1 Manipulating a DataTable

A data table can be manipulated by three operation

Selection

Updation

Deletion

SELECTION OF SPECIFIED ROWS

*use the select() method

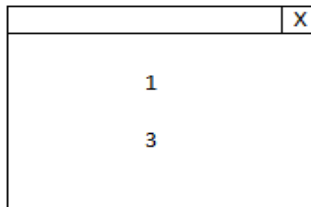
*parameter sent to select() is a string that contains some conditional operation

select rows with 'name=aaa' from the table 2.1 (student) and extract ID for that students.

code:-

```
string s1="name='aaa' ";
DataRow[]r1=t1.select(s1);
t1=>table reference
if(r1.Length==0)
    MessageBox.Show("norecords");
else
{
    string str=null;
    for(int i=0;i<r1.Length;i++)
    {
        DataRow t=r1[i];
        str+=t["ID"]+"\n";
    }
    MessageBox.Show(str);
}
```

o\p



1
3

UpDation:

Searches the DataTable student for all rows where name is equal to aaa.Once you identify these items, change the name from "aaa" to "ddd"

code:-

```
string s1="Name='aaa' ";
DataRow[] r1=t1.Select(s1);
for(int=0;i<r1.Length;i++)
{
    DataRow t=r1[i];
    t["name"]="ddd";
    r1[i]=t;
}
```

Deletion of Rows

- * use the Delete() method.
- * specify the index representing the row to remove.

Eg..

```
// delete the first row of the student table
t1.Rows[0].Delete();
t1.AcceptChanges();
```

5.6 DataView

- *View object is a stylized representation of a table.
- *DataView type allows you to programmatically extract a subset of data from a DataTable.
- *you can hold multiple views of same Table.

eg:-

- *create two DataViews.
View1 -> rows with name="aaa";
View2 -> rows with name="bbb";
*Bind those views to Data Grid (Container)

Source Code:

```
Dataview v1,v2;
Data Grid g1,g2;
v1=new DataVeiw(t1);
v2=new DataView(t1);
//t1 -> reference for student DataTable.
v1.RowFilter="name='aaa'";
v2.RowFilter="name='bbb'";
//RowFilter->propertyn gets or sets the expression used to filter which rows are viewed in DataView.
g1=new DataGrid();
g2=new DataGrid();
g1.Data Source=v1;
g2.Data Source=v2;
//g1->Contains View v1
//g2->Contains View v2
```

Members of DataView Type:

sort gets or sets the sort column or columns and sort order for the table.

Delete() Deletes a row at a sepcified index.

RowFilter already explained.

AddNew() Adds a new row to the DataView.

5.7 DataSet:

DataSet is an in-memory representation of any number of tables as well as any relationships between three tables and any constraints

Members of DataSet:

DataSetName: Represents the friendly name of this DataSet.

Relations: Gets the collection of relations that link tables and allows navigation from parent DataTables to child DataTables.

Tables : Provides access to the collection tables maintained by the DataSet.

Clear(): Completely clear the DataSet data.

Clone(): Clones the structure of the DataSet

GetChild Relations(): Returns the collection of child related that belong to the specified table.

Add Tables to DataSet:

```
Code: Data Set s1=new Data Set("Inventory");
      DataTable t1=new DataTable("customer");
      DataTable t2=new DataTable("orders");
//Add Tables to DataSet
      s1.Tables.Add(t1);
      s1.Tables.Add(t2);
```

Access Rows of the specified table:

```
s1.Tables["orders"].Rows[1];
//access 1st row of orders table of DataSet s1.
```

5.8 Data Relation:

*Once a DataSet has been populated with a number of tables, you can programmatically model their parent/child relationships.

*For a relationship to be established each table must have an identically named column of the same data type

DataSet s1:student,details

ID=>identically named column of same datatype

Creation of Relation:

```
Data Relation dr=new Data Relation("student
Details",s1.Tables["student"].columns["ID"],s1.Tables["details"].columns["ID"]);
s1.Relations.Add(dr);
```

Properties of DataRelation:

Obtains information about the child table in the relationship

Child Columns

ChildKeyConstraint

Child Table

obtain information about the parent table in the relationship

Parent Columns

ParentKeyConstraint

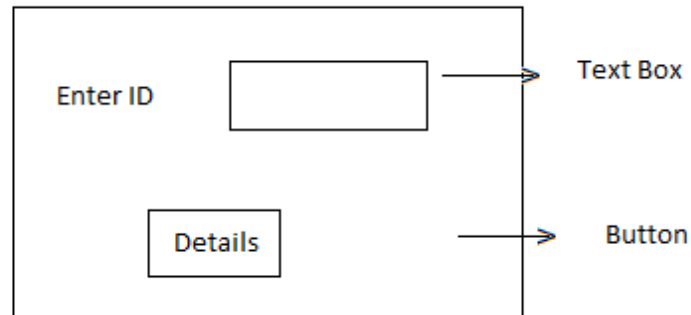
ParentTable

Relation Name Gets or sets the name of the relation

Navigation between Related Tables:-

By means of samples you will have a Data relation that allows you to

Design the form:-



While clicking button "Details" get name from the students table and Languages know from the Details Data table and display that using the message box.

sample code for eventhandler:-

```
public void b1-click(object sender,Event Args e)
{
string str=" ";
DataRow        r1=null;
DataRow[]      r2=null;
int c1=int .parse(t1.Text);
r1 =s1.Tables["student"].Rows[c1];
str=r1["name"];
r2=r1.GetChildRows(s1.Relations["studentDetails"])
foreach(Data Row d in r2)
str+=d["lang"];
Message Box.Show(str);
}
```

5.9 Data Provider

*Used to access data from Data store.

*Major Types:

*OleDb DataProvider

*SQL Data Provider

*OleDb DataProvider:-

Used to access data located in any data store that supports the classic OLEDB protocol

Namespace:-System.Data.OleDb;

*SQL Data provider:

used to access data from SQL server data stores.

Namespace:-System.Data.SqlClient

ADO.NET Providers:-

- 1.Microsoft.JET.OLEDB.4.0 connect to an access database
- 2.MSDAORA connect to an Oracle server
- 3.SQLOLEDB connect to the SQL server.

5.9 .1 OleDb Data Provider:

Members of System.Data.OleDb:-

- OleDbCommand Represents a SQL query command to be made to a data source
- OleDb Connection Represents an open connection to a data source
- OleDb DataReader provides a way of reading a forward-only stream of data records from a data source
- OleDb DataAdapter represents a set of data commands and a data connection used to fill and update the contents of a Data Set.

Create connections:-

```
OleDb Connection cn=new OleDb Connection();
cn.Connection String="Provider=SQLOLEDB.1;"+"User ID=sa;pwd;"+"@"Data Source=c: \sample;";
cn.Open();
cn.Close();
```

Members of the OleDb Connection:-

- connection string: Gets or sets the string used to open a session with a data store.
- Database : gets the name of the database maintained by the connection object.
- DataSource : Gets the location of the data base maintained by the connection object.
- Open() : opens a data base connection
- Provider : Gets the name of the provider maintained by the connection object.
- Close() : closes the connection to the datasource.

OleDb Command:-

By using OleDb Command class you can submit SQL queries to the database.

eg:-

```
//specify SQL command and connection as
//Constructor parameters.
String str="select*from student where name='aaa'";
OleDbCommand co=new OleDbCommand(str,cn);
//specifySQL command and connection via properties.
string str1="select *from student where name='aaa'";
OleDb Command co=new OleDb Command();
co.connection=cn;
co.commandText=str1
```

Members of OleDb Command Type:-

Command text	Gets or sets the SQL command text to run against the data source
Connection	Gets or sets the OleDb connection
ExecuteReader()	Returns an instance of an OleDb DataReader which provides forward only ,read only access to underlying data.
ExecuteNonQuery()	This method issues the command text to the data store without returning on OleDbDataReader type

Code to Access data from data store

```
Using System.Data;
Using System;
Class Data
{
    Public static void Main()
    {
        //Make connections
        OleDbConnections cn=new OleDbConnection();
        Cn.ConnectionString="Provider=SQLOLEDB.1";+"UserID
        =pa;Pwd"; +"DataSource=c:\Sample";

        Cn.Open();
        //create SQL Command
        String str="Select * from student";
        OleDbCommand co=new OleDbCommand(str,cn);
        //obtain DataReader via ExecuteReader()
        OleDbDataReader re;
        Re=co.ExecuteReader();
        //obtain the records through DataReader
        while(re.Read())
        {
            Console.WriteLine(re["ID"]);
        }
    }
}
```

5.9.2 OleDbDataReader

DataReaders are useful only when submitting SQL selection statements to underlying data store.

Program:

```
using System;
```

```

using System.Data.OleDb;
class Data
{
    //close the DataReader and connection
    re.close();
    cn.close();
}
}

```

Inserting ,updatingand deleting records using OleDbCommand

Insertion:

```

String str="Insert into Inventory"+"(name,ID)values"+"('aaa','123');
OleDbCommand co=new OleDbCommand(str,cn);
Co.ExecuteNonQuery();

```

Updation:

```

String str="Update student set name="bbb" where name='aaa'";
OleDbCommand co=new OleDbCommand(str,cn);
Co.ExecuteNonQuery();

```

Deletion:

```

String str="Delete from student where name='aaa'";
OleDbCommand co=new OleDbCommand(str,cn);
Co.ExecuteNonQuery();

```

Example

5.10 SqlProvider

Used to access data from SQL database.

Syntax for creating Sql Provider is similar to OleDb provider.**Instead of OleDb write Sql**

Eg: OleDbConnection->SqlConnection
 OleDbCommand ->SqlCommand

Example

```

using System;
using System.Data;
using System.Data.SqlClient;

class MainClass
{
    static void Main(string[] args)
    {

```



```
string connString = "server=(local)\\SQLEXPRESS;database=MyDatabase;Integrated Security=SSPI;";

string sql = @"select * from employee";

SqlConnection conn = null;
SqlDataReader reader = null;

try
{
    conn = new SqlConnection(connString);
    conn.Open();

    SqlCommand cmd = new SqlCommand(sql, conn);
    reader = cmd.ExecuteReader();

    Console.WriteLine("Querying database {0} with query {1}\n", conn.Database, cmd.CommandText);

    while(reader.Read()) {
        Console.WriteLine("{0} | {1}", reader["FirstName"].ToString().PadLeft(10), reader[1].ToString().PadLeft(10));
    }
}catch (Exception e)
{
    Console.WriteLine("Error: " + e);
}
finally
{
    reader.Close();
    conn.Close();
}
}
```